Filoo — Fichier Circulaire
Rapport Préliminaire

Clément Dudouet

25 mars 2007

Table des matières

1 Introduction **2**
1.1 Qu'est ce `filoo` ? . . . 2
1.2 Utilité de `filoo` . . . 2

2 Utilisation **2**
2.1 Fonction `filoo_open` . . . 3
2.2 Fonction `filoo_close` . . . 3
2.3 Fonction `filoo_create` . . . 4
2.4 Fonction `filoo_read` . . . 6
2.5 Fonction `filoo_write` . . . 8
2.6 Fonction `filoo_lseek` . . . 9

3 Spécifications des choix **10**
3.1 Lecture et écriture simultannées . . . 10
3.2 Autres cas . . . 10

4 Tests de recette **10**
4.1 Test des codes d'erreurs . . . 10
Erreurs de `filoo_create` . . . 10
Erreurs de `filoo_open` . . . 10
Erreurs de `filoo_close` . . . 10
Erreurs de `filoo_read` . . . 10
Erreurs de `filoo_write` . . . 10
4.2 Tests de base . . . 11
Création d'un fichier . . . 11
Lecture avec offset . . . 11
4.3 Tests élaborés . . . 11
Ecriture d'un nombre d'octets supérieur à la taille du fichier . 11
4.4 Concurrence . . . 11
Plusieurs lecteurs . . . 11
Lecteur et Ecrivain . . . 11
Ecrivains multiples . . . 11

5 Implémentation **12**
5.1 `filoo_create` . . . 12
5.2 `filoo_open` . . . 12
5.3 `filoo_close` . . . 12
5.4 `filoo_read` . . . 12
5.5 `filoo_write` . . . 13
5.6 `filoo_lseek` . . . 13

1 Introduction

1.1 Qu'est ce `filoo` ?

`Filoo` – pour *File Loop* – est une API implémentant des fichiers circulaires sous un sytème d'exploitation Linux.

Le principe est simple :
- On créé un fichier dont on spécifie une taille fixe.
- On utilise ce fichier comme n'importe quel autre fichier texte à la différence près que lors de l'écriture, lorsque la fin de fichier est atteinte, l'écriture continue en début de fichier.
- Lors de l'ouverture du fichier, le pointeur ne se trouve pas au début du fichier mais à l'endroit de la donnée la plus ancienne. De ce fait, l'API est totalement opaque vis à vis de l'utilisateur. La lecture (ou l'écriture) commence à l'endroit où le pointeur a été placé au moment de l'ouverture du ficher. Ainsi, il est semblable à n'importe quel autre fichier.

Cette implémentation sera effectuée en utilisant un principe d'entête de fichier. C'est à dire que tout fichier `filoo` sera un fichier régulier auquel on aura ajouté un entête contenant les informations nécessaires à la bonne utilisation de celui-ci.

1.2 Utilité de `filoo`

La principale utilité de cette API concerne les fichiers *log*.
En effet, grâce à cette API, on spécifie une fois pour toutes la taille désirée des fichiers circulaires `filoo`, et celle-ci ne sera jamais modifiée, donc jamais dépassée. On peut ainsi éviter les situations embarassantes de *Disk Full*.

2 Utilisation

Certains appels système, utilisés pour la lecture et l'écriture dans les fichiers réguliers, nécessitent une ré-implémentation pour les fichiers circulaires `filoo`, tandis que d'autres seront utilisés normalement comme pour les fichiers réguliers.

Voici les pages de manuel (MAN) des principales fonctions ré-implémentées de cette API :

2.1 Fonction `filoo_open`

`filoo_open(2)` `Manuel d'utilisateur` `filoo_open(2)`

```
NOM
       filoo_open - Ouverture d'un fichier circulaire filoo existant

SYNOPSIS
       #include <unistd.h>

       int filoo_open(const char *pathname, int flags);

DESCRIPTION
       L'appel  système  filoo_open()  sert a convertir un chemin d'accès en
       descripteur de fichier.  Le nouveau descripteur  de  fichier  est  con-
       figuré  pour  rester  ouvert  au  travers  des  fonctions  exec  (voir
       fntl(2)).   Le  pointeur  de  position  dans  le  fichier  est   placé
       au niveau de la donnée la plus ancienne contenue dans le fichier.

       Le paramètre flags est l'un des éléments suivants O_RDONLY, O_WRONLY
       ou O_RDWR qui réclament respectivement l'ouverture du fichier en  lec-
       ture seule, écriture seule, ou lecture-écriture.

ERREURS
       EINCOHERENT
              pathname désigne un fichier incohérent au  sens  des  fichiers
              circulaires (l'entête est incohérent).

       EEXIST pathname ne correspond pas a un fichier existant

       EACCES L'accès  demandé  au fichier est interdit, ou la permission de
              parcours pour l'un  des  répertoires  du  chemin  pathname  est
              refusée, ou le fichier n'existe pas.

       ...    cf.  open(2)

VOIR AUSSI
       filoo_close(2), filoo_create(2), filoo_read(2), filoo_write(2)

AUTEUR
       Clément DUDOUET

DISPONIBILITE
       Le projet est disponible a cette adresse :

       http://code.google.com/p/filoo

       Et par l'intermédiaire de SVN en utilisant la commande :

       svn checkout http://filoo.googlecode.com/svn/trunk/ filoo


version 1.0                   Fevrier 11, 2007                 filoo_open(2)
```

2.2 Fonction `filoo_close`

```
filoo_close(2)              Manuel d'utilisateur              filoo_close(2)


NOM
       filoo_close  - Fermeture d'un descripteur de fichier correspondant a un
       fichier circulaire filoo
```

```
SYNOPSIS
       #include <unistd.h>

       int filoo_close(int fd);

DESCRIPTION
       filoo_close ferme le descripteur de fichier fd de manière a  ce  qu'il
       ne  référence  plus  aucun fichier et puisse donc être réutilisé a
       d'autres fins.

       filoo_close écrit de plus dans l'entête du fichier, si  celui-ci  est
       ouvert  en écriture, la position du pointeur de lecture/écriture afin
       que lors de la prochaine  ouverture  ce  dernier  soit  placé  sur  la
       donnée la plus ancienne.

VALEUR RENVOYEE
       filoo_close  renvoie 0 s'il réussit, ou -1 en cas d'échec, auquel cas
       errno contien le code d'erreur correspondant.

ERREURS
       EPTPOSITION
              Impossible d'écrire dans l'entête la position du pointeur afin
              que celui-ci référence la donnée la plus ancienne

       EBADFD Le descripteur de fichier fd est invalide

VOIR AUSSI
       filoo_create(2), filoo_open(2), filoo_read(2), filoo_write(2)

AUTEUR
       Clément DUDOUET

DISPONIBILITE
       Le projet est disponible a cette adresse :

       http://code.google.com/p/filoo

       Et par l'intermédiaire de SVN en utilisant la commande :

       svn checkout http://filoo.googlecode.com/svn/trunk/ filoo



version 1.0                    Fevrier 11, 2007                 filoo_close(2)
```

2.3 Fonction `filoo_create`

```
filoo_create(2)              Manuel d'utilisation              filoo_create(2)



filoo_create - Création immédiate d'un fichier circulaire de taille fixée

SYNOPSIS
       #include <unistd.h>

       int filoo_create(const char *pathname, unsigned long size, int  mode);

DESCRIPTION
       L'appel  système  filoo_create()  sert  a créer un fichier circulaire
       filoo de chemin pathname et a convertir ce chemin d'accès en  descrip-
       teur de fichier.  Le nouveau descripteur de fichier est configuré pour
```

rester ouvert au travers des fonctions exec (voir fntl(2)). Le pointeur de position dans le fichier est placé au début de celui-ci.

La création est immédiate. C'est a dire que, lors de cet appel, le ficher est créé et rempli de zéros afin qu'il atteigne la taille désirée.

Le paramètre size est un entier strictement supérieur a zéro décrivant la taille du fichier (en kilo octets) circulaire filoo a créer. Afin que le fichier circulaire soit stocké sur un nombre de pages entier, on arrondira ce paramètre au multiple de 4ko le plus proche. Cette taille restera constante tout au long de la vie de ce fichier.

Le paramètre mode indique les permissions a utiliser si un nouveau fichier est créé. Cette valeur est modifiée par le umask du processus : la véritable valeur utilisée est (mode & ~umask). Notez que ce mode ne s'applique qu'aux accès ultérieurs du fichier nouvellement créé. L'appel open qui crée un fichier dont le mode est en lecture seule fournira quand même un descripteur de fichier en lecture et écriture.

Les constantes symboliques suivantes sont disponibles pour mode :

S_IRWXU
: 00700 L'utilisateur (propriétaire du fichier) a les autorisations de lecture, écriture, exécution.

S_IRUSR (S_IREAD)
: 00400 L'utilisateur a l'autorisation de lecture.

... : cf. open(2).

VALEUR RENVOYEE

filoo_create renvoie le nouveau descripteur de fichier si la création a réussi ou -1 si elle a échoué, auauquel cas errno contient le code d'erreur.

ERREURS

EBADSIZE
: size passé en argument est invalide.

ENOSPC pathname devrait être créé mais le périphérique concerné n'a plus assez de place pour un nouveau fichier.

ETOOBIG
: pathname devrait être créé, le périphérique concerné a assez de place pour la création d'un fichier, mais pas assez pour la taille de fichier désirée.

EEXIST pathname existe déja

... : cf. open(2)

VOIR AUSSI

filoo_open(2), filoo_close(2), filoo_read(2), filoo_write(2)

AUTEUR

Clément DUDOUET

DISPONIBILITE

Le projet est disponible a cette adresse :

http://code.google.com/p/filoo

Et par l'intermédiaire de SVN en utilisant la commande :

svn checkout http://filoo.googlecode.com/svn/trunk/ filoo

version 1.0 11 fevrier 2007 filoo_create(2)

2.4 Fonction `filoo_read`

filoo_read(2) Manuel d'utilisateur filoo_read(2)

NOM

filoo_read - Lecture dans un fichier circulaire préalablement ouvert

SYNOPSIS

```
#include <sys/types.h>
#include <unistd.h>

ssize_t filoo_read(int fd, void *buffer, size_t count);
```

DESCRIPTION

filoo_read lit jusqu'a count octets depuis le pointeur du descripteur de fichier fd dans le tampon pointé par buf. Ce fichier doit être un fichier circulaire préalablement ouvert via l'appel système filoo_open(2) ou filoo_create(2).

Si count vaut zéro, filoo_read renvoie zéro et n'a pas d'autres effets. Si count est supérieur a SSIZE_MAX, le résultat est indéfini.

filoo_read se distingue de l'appel système read en ce sens qu'il a une lecture "circulaire". C'est a dire que lorsque la fin du fichier est atteinte, la lecture continue au début de celui-ci. Autrement dit, la lecture du n-ème octet correspond en réalité a la lecture de l'octet n [size] (sachant que l'entête spécifiant le fichier filoo, ne peut être lue par filoo_read).

VALEUR RENVOYEE

filoo_read renvoie -1 s'il échoue, auquel cas errno contient le code d'erreur correspondant, et la position de la tête de lecture est indéfinie. Dans les autres cas, filoo_read renvoie le nombre d'octets lus et avance la tête de lecture de ce nombre.

ERREURS

EBADF fd n'est pas un descripteur valide, ou n'est pas en lecture.

... cf. read(2)

VOIR AUSSI

filoo_close(2), filoo_create(2), filoo_open(2), filoo_write(2)

AUTEUR

Clément DUDOUET

DISPONIBILITE

Le projet est disponible a cette adresse :

http://code.google.com/p/filoo

Et par l'intermédiaire de SVN en utilisant la commande :

svn checkout http://filoo.googlecode.com/svn/trunk/ filoo

2.5 Fonction `filoo_write`

```
filoo_write(2)              Manuel d'utilisateur              filoo_write(2)


NOM
       filoo_write - Ecriture dans un fichier circulaire préalablement ouvert

SYNOPSIS
       #include <unistd.h>

       ssize_t filoo_write(int fd, const void *buffer, size_t count);

DESCRIPTION
       filoo_write écrit count bytes dans  le  fichier  référencé  par  le
       descripteur de fichier fd depuis le tampon commençant en buffer.

       Le  descripteur de fichier fd doit correspondre a un fichier circulaire
       préalablement  ouvert  grâce  aux  appels  sytème  filoo_open(2)  ou
       filoo_create(2)

       Cet  appel  système agit comme l'appel write(2) a la différence près
       que l'écriture du (size + n) ème octet, écrase l'octet n.

VALEUR RENVOYEE
       filoo_write  renvoie  le  nombre  d'octets écrits (0 signifiant aucune
       écriture), ou -1 s'il échoue,  auquel  cas  errno  contient  le  code
       d'erreur correspondant.

ERREURS
       EBADSEEK
              Si le pointeur d'écriture a été placé dans la zone d'entête
              du fichier, l'écriture est interdite.

       EBADFD fd n'est pas un descripteur de  fichier  valide,  ou  n'est  pas
              ouvert en écriture.

       ...    cf.  write(2)

VOIR AUSSI
       filoo_close(2), filoo_create(2), filoo_open(2), filoo_read(2)

AUTEUR
       Clément DUDOUET

DISPONIBILITE
       Le projet est disponible a cette adresse :

       http://code.google.com/p/filoo

       Et par l'intermédiaire de SVN en utilisant la commande :

       svn checkout http://filoo.googlecode.com/svn/trunk/ filoo


version 1.0                  Fevrier 11, 2007               filoo_write(2)
```

2.6 Fonction `filoo_lseek`

La fonction `filoo_lseek` agit de la même manière que la fonction `lseek` à la différence près que l'entête du fichier ne peut être atteint.

```
filoo_lseek(2)              Manuel d'utilisateur              filoo_lseek(2)



NOM
       filoo_lseek - Positionner la tête de lecture/écriture dans un fichier
       circulaire filoo

SYNOPSIS
       #include <sys/types.h>
       #include <unistd.h>

       off_t filoo_lseek(int fd, off_t offset, int whence);

DESCRIPTION
       filoo_lseek place la tête de lecture/écriture a  la  position  offset
       dans  le  fichier  associé  au  descripteur fd en suivant la directive
       whence (cf.  lseek )

       De plus, cette fonction empêche le pointeur de position d'être placé
       dans l'entête du fichier filoo ou après la fin de ce fichier.

VALEUR RENVOYEE
       filoo_lseek  s'il  réussit,  renvoie le nouvel emplacement, mesuré en
       octets depuis le début du fichier. En cas d'échec la valeur (off_t)-1
       est renvoyée, et errno contient le code d'erreur correspondant.

ERREURS
       EEOF   La  fin  du  fichier  serait  dépassée en cas de changement de
              position du pointeur.

       ...    cf.  lseek(2)

VOIR AUSSI
       filoo_close(2),    filoo_create(2),    filoo_open(2),    filoo_read(2),
       filoo_write(2)

AUTEUR
       Clément DUDOUET

DISPONIBILITE
       Le projet est disponible a cette adresse :

       http://code.google.com/p/filoo

       Et par l'intermédiaire de SVN en utilisant la commande :

       svn checkout http://filoo.googlecode.com/svn/trunk/ filoo



version 1.0                      26 mars 2007                  filoo_lseek(2)
```

3 Spécifications des choix

3.1 Lecture et écriture simultannées

Dans le cas de lecture et écritures simultannées, le problème du positionnement du pointeur dans le fichier se pose.
En effet, l'écriture va modifier les données pointées par la position de lecture du programme effectuant la lecture du fichier.
Afin d'éviter ce problème, avant toute opération de lecture ou écriture, nous vérifierons la position du pointeur.
Pour ce faire, à chaque modification effectuée par une écriture dans le fichier, nous mettrons à jour cette position dans l'inode correspondant au fichier afin que les autres programmes puissent s'y référer.

Ainsi avant toute opération impliquant la position dans le fichier, nous vérifierons que celle-ci n'a pas été modifiée.

3.2 Autres cas

Il en va de même pour tous les autres cas de figure possibles.

4 Tests de recette

4.1 Test des codes d'erreurs

Erreurs de `filoo_create`

- Création d'un fichier de taille négative
- Création d'un fichier de taille trop grande pour la capacité de stockage
- Création d'un fichier déjà existant

Erreurs de `filoo_open`

- Fichier non existant
- Fichier dont l'entête est incohérent

Erreurs de `filoo_close`

- Test de fermeture avec un mauvais file descriptor

Erreurs de `filoo_read`

- Mauvais descripteur de fichier

Erreurs de `filoo_write`

- Mauvais descripteur de fichier

4.2 Tests de base

Création d'un fichier

Création d'un fichier d'une taille donnée.
Ecriture du même motif dans l'intégralité du fichier.

Lecture avec offset

Lecture d'un nombre d'octets donné d'un fichier existant en partant d'un offset spécifié en argument.

4.3 Tests élaborés

Ecriture d'un nombre d'octets supérieur à la taille du fichier

Dans un fichier existant, écriture d'un nombre d'octets tel que l'on boucle au début du fichier.
Lecture de ce fichier afin de s'assurer du bon fonctionnement.

4.4 Concurrence

Plusieurs lecteurs

Lecture d'un même fichier `filoo` par plusieurs lecteurs simultanément.

Lecteur et Ecrivain

- Ouverture du fichier `filoo` par un lecteur et un écrivain simultanément
- Ecriture de k octets par l'écrivain dans le fichier
- Lecture par le lecteur de n octets
- Ecriture de m octets
- Lecture

Ce test doit nous permettre de vérifier que le pointeur du lecteur est bien déplacé lorsqu'il y a écriture.

Ecrivains multiples

- Ouverture du même fichier circulaire par plusieurs programmes
- Ecriture par l'un des programmes de k octets
- Ecriture de n octets par un programme différent

Ce test doit nous permettre de vérifier que le pointeur d'écriture est bien déplacé par tout autre programme ayant précédemment écrit.
L'écriture des différents programmes doit donc être séquentielle.

5 Implémentation

Voici les détails correspondants à l'implémentation des fonctions précédemment décrites.

5.1 `filoo_create`

1. Vérification des arguments
2. Création du fichier à l'aide de l'appel système `create`
3. Écriture de l'entête contenant les informations : "FILOO", taille, position du pointeur
4. Remplissage immédiat de l'intégralité du fichier de sorte qu'il ait la taille désirée
   Dans le cas d'un manque d'espace afin de remplir le fichier, on le supprime complètement du disque et on renvoie l'erreur correspondante.
5. Renvoi du file descriptor correspondant

5.2 `filoo_open`

1. Ouverture du fichier avec l'appel système `open`
2. Vérification de l'entête du fichier :
   - Type de fichier `filoo`
   - Taille du fichier > 0
   - Position du pointeur
3. Si les vérification échouent, fermeture du fichier et renvoi du code d'erreur correspondant
4. Sinon on place le pointeur de position à l'endroit spécifié dans l'entête
5. On retourne le descripteur de fichier ainsi obtenu.

5.3 `filoo_close`

1. Vérification que le file descriptor fourni en argument correspond bien à un fichier de type `filoo`
2. Mise à jour de la position du pointeur
3. Ecriture de cette position dans l'entête
4. Fermeture du fichier à l'aide de l'appel système `close`

5.4 `filoo_read`

1. Vérification des arguments :
   - fd correspond bien à un fichier `filoo`
   - offset positif

- buffer valide

2. <**Section Critique**> Mise à jour du pointeur de position du fichier par lecture de l'inode </**Section Critique**>
3. Lecture du nombre d'octets spécifiés et stockage dans le buffer passé en argument.
4. On retourne le nombre de caractères lus

5.5 `filoo_write`

1. Vérification des arguments :
   - fd correspond bien à un fichier `filoo`
   - nombre de caractères positif
   - buffer valide
2. <**Section Critique**> Mise à jour du pointeur de position du fichier par lecture de l'inode </**Section Critique**>
3. Ecriture des n premiers caractères du buffer dans le fichier circulaire correspondant à fd
4. Ecriture de la nouvelle position du pointeur du fichier circulaire dans l'inode.
5. Renvoi du nombre de caractères écrits dans le fichier circulaire

5.6 `filoo_lseek`

1. Vérification des arguments :
   - fd correspond bien à un fichier `filoo`
   - offset non négatif
   - whence valide
2. - Si `whence` vaut `SEEK_SET` alors on place le pointeur de position au caractère `(offset + taille_entête)` du fichier.
   - Si `whence` vaut `SEEK_END` alors on place le pointeur à la fin du fichier et on termine
   - Si `whence` vaut `SEEK_CUR` et que `(position_courante + offset)` dépasse la fin du fichier alors on termine en renvoyant une erreur.
3. On renvoie la position du pointeur dans le fichier (numéro de caractère en partant de la fin de l'entête)